Testing and Verification of VLSI
EE709

Mohit
20D070052

April 2023

1 Student Details

Name: Mohit
Roll No: 20D070052

2 Question - Analysis of 4 bit Adder

2.1 First Part

Consider the following subset A of the domain: those combinations of x and y such that an odd number of bits of x and an odd number of bits of y are 1. Using the BDD package, find the image of the set A in the range.

To find the image of the set A in the range, we first need to define the function that maps inputs x and y to the range. Let's assume that our function f(x, y) simply returns the sum of x and y.

2.1.1 Calling C Code

Given below is the C code for calling the BDD Package:

```
#include<stdlib.h>
#include<stdio.h>
#include<bdduser.h>

int main(int argc, char* argv[]){
bdd_manager bddm = bdd_init();
bdd X[8], S[4], C, Y[5], kai, Q[9];
int i, asc;
for(i=0; i<8; i++){
X[i] = bdd_new_var_last(bddm);
}
for(i=0; i<5; i++){
Y[i] = bdd_new_var_last(bddm);
}

C = bdd_zero(bddm);

for(i=0; i<4; i++){
S[i] = bdd_xor(bddm, bdd_xor(bddm, X[i], X[i+4]), C);
C = bdd_or(bddm, bdd_and(bddm, C, bdd_or(bddm, X[i], X[i+4])), bdd_and(bddm, X[i],
```

```
}

kai = bdd_one(bddm);
for(i=0; i<4; i++)
kai = bdd_and(bddm, kai, bdd_xnor(bddm, S[i], Y[i])); //Characteristic function to
kai = bdd_and(bddm, kai, bdd_xnor(bddm, C, Y[4]));

bdd x_space = bdd_zero(bddm);
for(i=0; i<4; i++)
x_space = bdd_xor(bddm, x_space, X[i]);
bdd x2_space = bdd_zero(bddm);
for(i=4; i<8; i++)
x2_space = bdd_xor(bddm, x2_space, X[i]);
x_space = bdd_and(bddm, x_space, x2_space); //XOR of both x1x2x3x4 and x5x6x7x8 sh

for(i=0; i<8; i++){
Q[i] = X[i];
}
Q[8] = 0;

asc = bdd_new_assoc(bddm, Q, 0);
bdd_assoc(bddm, asc);

bdd y_img = bdd_exists(bddm, bdd_and(bddm, x_space, kai));

printf("Question 1: Image of odd number of high bits in both inputs and output\n\n
printf("Image:\n");
bdd_print_bdd(bddm, y_img, NULL, NULL, NULL, stdout);
```

2.1.2 BDD Result

The resultant BDD is as shown below:

```
Image:
if var.8
  if var.9
```

```
    if var.10
      if var.11
       !var.12
      else if !var.11
       0
      endif var.11
     else if !var.10
       1
     endif var.10
   else if !var.9
     if var.10
      0: if var.11
      !var.12
      else if !var.11
      1
      endif var.11
     else if !var.10
      1: if var.11
      1
      else if !var.11
      var.12
      endif var.11
     endif var.10
   endif var.9
else if !var.8
if var.9
if var.10
subformula 0
else if !var.10
1
endif var.10
else if !var.9
if var.10
1
else if !var.10
subformula 1
endif var.10
endif var.9
```

```
endif var.8
```

2.2 Second Part

Consider the following subset B of the range: the set of all 5-bit numbers
c y3 y2 y1 y0
such that the number of bits in the number is odd. Using the BDD package, find the pre-image of the set B in the domain.

To find the pre-image of a set B in the domain, we need to construct a BDD that represents the Boolean function that defines the set B, and then use the BDD to compute the pre-image.

2.2.1 Calling C code

Given below is the C code for calling the BDD Package:

```
bdd y_space = bdd_zero(bddm);
for(i=0; i<4; i++)
y_space = bdd_xor(bddm, y_space, Y[i]); //XOR of y1y2y3y4y5 should be 1

for(i=0; i<5; i++){
Q[i] = Y[i];
}
Q[5] = 0;

asc = bdd_new_assoc(bddm, Q, 0);
bdd_assoc(bddm, asc);

bdd x_preimg = bdd_exists(bddm, bdd_and(bddm, y_space, kai));

printf("Pre-Image:\n");
bdd_print_bdd(bddm, x_preimg, NULL, NULL, NULL, stdout);
```

2.2.2 BDD Result

Given Below is the result pre-image using BDD package.

```
Pre-Image:
if var.0
  if var.1
    if var.2
      if var.3
        0: if var.4
          1: if var.5
            2: if var.6
              var.7
            else if !var.6
              !var.7
            endif var.6
          else if !var.5
            !subformula 2
          endif var.5
        else if !var.4
          3: if var.5
            subformula 2
          else if !var.5
            var.7
          endif var.5
        endif var.4
      else if !var.3
        !subformula 0
      endif var.3
    else if !var.2
      if var.3
        4: if var.4
          5: if var.5
            !var.7
          else if !var.5
            var.7
          endif var.5
```

```
        else if !var.4
          6: if var.5
            !var.7
          else if !var.5
            subformula 2
          endif var.5
        endif var.4
      else if !var.3
        !subformula 4
      endif var.3
    endif var.2
  else if !var.1
    if var.2
      if var.3
        7: if var.4
          !subformula 3
        else if !var.4
          !subformula 5
        endif var.4
      else if !var.3
        !subformula 7
      endif var.3
    else if !var.2
      if var.3
        8: if var.4
          !subformula 6
        else if !var.4
          subformula 1
        endif var.4
      else if !var.3
        !subformula 8
      endif var.3
    endif var.2
  endif var.1
else if !var.0
  if var.1
    if var.2
      if var.3
```

```
      9: if var.4
        subformula 3
      else if !var.4
        !subformula 3
      endif var.4
    else if !var.3
      !subformula 9
    endif var.3
  else if !var.2
    if var.3
      10: if var.4
        subformula 6
      else if !var.4
        !subformula 6
      endif var.4
    else if !var.3
      !subformula 10
    endif var.3
  endif var.2
else if !var.1
  if var.2
    if var.3
      11: if var.4
        !subformula 5
      else if !var.4
        subformula 5
      endif var.4
    else if !var.3
      !subformula 11
    endif var.3
  else if !var.2
    if var.3
      12: if var.4
        subformula 1
      else if !var.4
        !subformula 1
      endif var.4
    else if !var.3
```

```
        !subformula 12
      endif var.3
    endif var.2
  endif var.1
endif var.0
```

2.3 Third Part

Lets prove a property about a four bit adder: show (using BDD's) that every even 4-bit number can be expressed as a sum of two prime numbers.

For this we need to prove that both the BDD's for even numbers and intersection of even numbers and the image of prime numbers are same.

2.3.1 Calling C code

Firstly given below is the C code for calling the BDD package:

```
bdd prime_space = bdd_one(bddm);
bdd pi1, pi2, pi3, pi4; //Taking all prime implicants for prime numbers
for(i=0; i<1; i++){
pi1 = bdd_and(bddm, X[4*i], bdd_and(bddm, X[4*i+1], bdd_not(bddm, X[4*i+3]))); //x
pi2 = bdd_and(bddm, X[4*i+1], bdd_and(bddm, bdd_not(bddm, X[4*i+2]), bdd_not(bddm,
pi3 = bdd_and(bddm, X[4*i], bdd_and(bddm, X[4*i+2], bdd_not(bddm, X[4*i+1]))); //x
pi4 = bdd_and(bddm, X[4*i], bdd_and(bddm, X[4*i+1], bdd_not(bddm, X[4*i+2]))); //x

prime_space = bdd_and(bddm, prime_space, bdd_or(bddm, bdd_or(bddm, pi1, pi2), bdd_
}

bdd y_even = bdd_and(bddm, bdd_and(bddm, bdd_not(bddm, Y[4]), bdd_or(bddm, Y[3], Y

for(i=0; i<8; i++){
Q[i] = X[i];
}
```

```
Q[8] = 0;
asc = bdd_new_assoc(bddm, Q, 0);
bdd_assoc(bddm, asc);

bdd y_prime_img = bdd_exists(bddm, bdd_and(bddm, prime_space, kai));

printf("Question 3: Proof of same BDD");

printf("-----------------------------------------------------\n");
printf("BDD for the image of prime numbers\n");
bdd_print_bdd(bddm, y_prime_img, NULL, NULL, NULL, stdout);
printf("-----------------------------------------------------\n");
printf("BDD for even numbers\n");
bdd_print_bdd(bddm, y_even, NULL, NULL, NULL, stdout);
printf("-----------------------------------------------------\n");
printf("BDD for the intersection between the even numbers and the image of prime r
bdd_print_bdd(bddm, bdd_and(bddm, y_even, y_prime_img), NULL, NULL, NULL, stdout);
printf("-----------------------------------------------------\n");

if(bdd_and(bddm, y_even, y_prime_img) == y_even){
printf("Both BDDs are the same, hence, the even numbers are a subset of the image
}
else{
printf("Error- disproven.\n");
}
return 0;
}
```

2.3.2 BDD for image of prime numbers

Given below is the result for the obtained BDD's for image of prime numbers.

```
if var.8
  if var.9
        0: if var.10
          1: if var.11
              !var.12
```

```
          else if !var.11
              1
            endif var.11
        else if !var.10
            1
          endif var.10
        else if !var.9
          if var.10
            subformula 1
          else if !var.10
            2: if var.11
             1
            else if !var.11
             var.12
         endif var.11
         end if var.10
  end if var.9
else if !var.8
      if var.9
       subformula 0
      else if !var.9
          if  var.10
           1
          else if !var.10
           subformula 2
          endif var.10
      endif var.9
endif var.8
```

2.3.3 BDD for even numbers

Given below is the result for the obtained BDD's for even numbers.

```
if var.8
0
```

```
else if !var.8
if  var.10
!var.12
else if !var.10
if var.11
!var.12
else if !var.11
0
endif var.11
endif var.10
endif var.8
```

2.3.4 BDD for intersection of both

Given below is the result for the obtained intersection BDD for image of prime numbers and even numbers.

```
if var.8
0
else if !var.8
if  var.10
!var.12
else if !var.10
if var.11
!var.12
else if !var.11
0
endif var.11
endif var.10
endif var.8
```

2.3.5 Result

As we can see that both the BDD's are exactly same and since it is a canonical form of representation, Therefore, we have proven that every even 4-bit

number can be expressed as a sum of two prime numbers using BDDs.

3 Conclusion

Hereby the assignment is completed using the BDD Package.